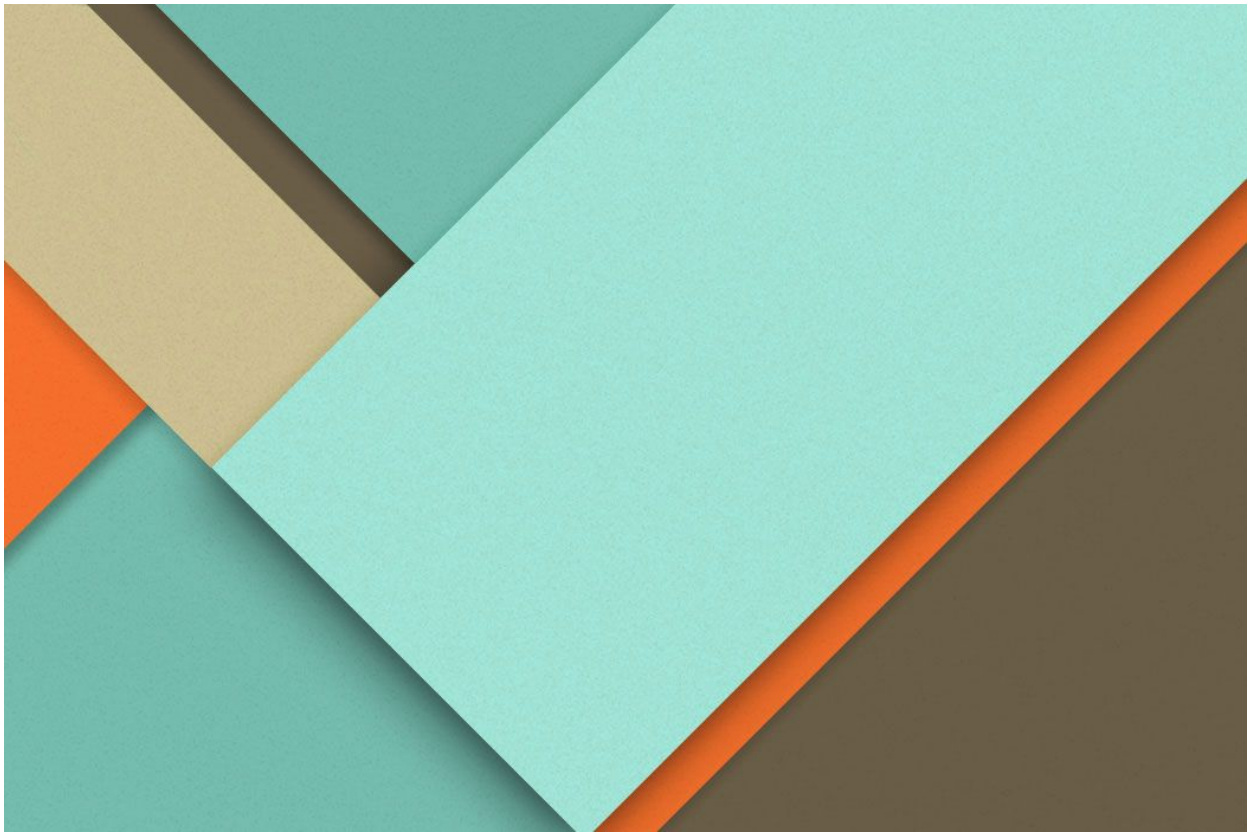


Phase 1

09.04.2016

Adonias Payan Lopez & Henna Girish Gohil
Team: Nexus

Nexus Messenger

Application Properties

We are developing a messenger app that allows a user to send messages to a single recipient. The messenger app will be developed under the Android JDK and we will be our RESTful web server for our backend/server. We will be using Signal Protocol for authentication. OpenSSL will be used to implement Transport Layer Security 1.2 (TLS) to handle the security layer and establishes connection between the server and client. We will be using OpenPGP to encrypt our messages. Furthermore, we will use MySQL database to store credentials of the users and messages. LetsEncrypt will be used to create key pairs to authorize certificates being passed.

Assets and Stakeholders

The stakeholder for this project will be the user of the app. The asset of the stakeholder is the message being sent through the server to another client (recipient).

Adversarial Model

- Outside Attacks:
    - We will incorporate end-to-end encryption to combat eavesdropping from outside attackers.
    - To limit the effectiveness of a brute force attack, the private-key size is going to be set to 2048-bit.
- Inside Attacks:
    - We will use endpoint authentication to combat Man-in-the-Middle (MitM) attacks. OpenSSL will allow us to use TLS to implement an authentication process for both messengers using mutual certificate authority.

Possible Vulnerabilities

Possible vulnerabilities of our system can consist of a workstation hijacking and DDOS attacks. Our app cannot guarantee security from someone stealing the device and gaining the messages that have been decrypted from the client side. In addition, no mechanisms will be set up to protect the dignity and availability of the server. The Signal protocol does not provide anonymity preservation and it requires servers for the relaying of the messages and storing of public key material.

Possible Related Previous Work

WhatsApp Encryption Overview, https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf

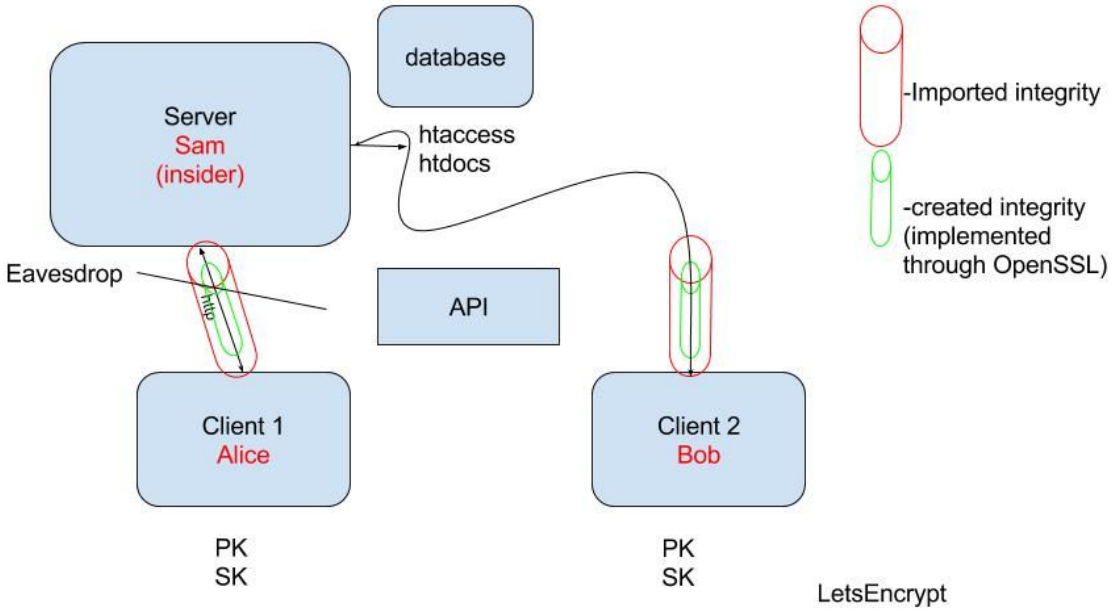
Use Cases/ Diagrams

Use Cases:

If user does not yet exist in the database, they must first register by inputting their desired username and password to interact with the server. This way we can securely store their credentials with MySQL. When registering, the username and password will go through the (green tunnel) to undergo encryption and the encrypted information will stored into the database (MySQL). Once, the registration process is complete, the user will then be free to send a message to another existing client in the database.

When sending a message, it will undergo encryption through OpenPGP and sent through the server. Once the connection with the server is established, the user's message can go to the recipient of the message. Before the recipient can receive the message, the user must GET the authentication certificate provided by the server. The public keys from both clients will be exchanged once a secured connection is established. When the message is being sent from the user, it is POSTed to the server. Then, the recipient GETs the message from the server. The public keys will be used to encrypt the relaying messages between the two users and upon receiving the message, the user's private key will be used to decrypt the message.

Project:



Complete Description of Solution

For our project, we will be using the android API, due to how there are built in distinct components, which can maximize security. Since android is an open source platform, the security is strengthened by its community of android users. Android protects applications and user data, system resources including network, and provides application isolation from the system, and other applications. RESTful will create a communication line between the server and the client using HTTP verbs, such as GET and POST. OpenSSL toolkit will utilize TLS that provides cryptographic functionality to secure the web server. We will be using Signal Protocol from Open Whisper Systems, which will help us setup a connection between the client and the server (red tunnel). The green tunnel will be provided by OpenPGP (Pretty Good Privacy) with a public and private key pair. To keep the client's information for future references, MySQL will be used to store the information after going through an encryption process. We will be using Signal as our protocol which provides confidentiality, integrity, participant consistency, authentication, forward and backward secrecy, causality and asynchronous communication.

LetsEncrypt will provide the certificates for the HTTPs server to make sure it's secure for the client.

Full Rigorous Analysis

We are using Android JDK because of how the open platform requires strong architecture and rigorous security programs. The android design allows us to be more flexible and supports an open platform while still protecting users of the platform. Android security had released a tool for testing SSL, which helps us developers find potential security issues on our application.

The reason why we chose to make a RESTful server is protect HTTP methods and create a secure HTTPS instead of HTTP. Creating a RESTful API for our server protects almost every aspect of client-to-server communication and it secures the integrity of the server.

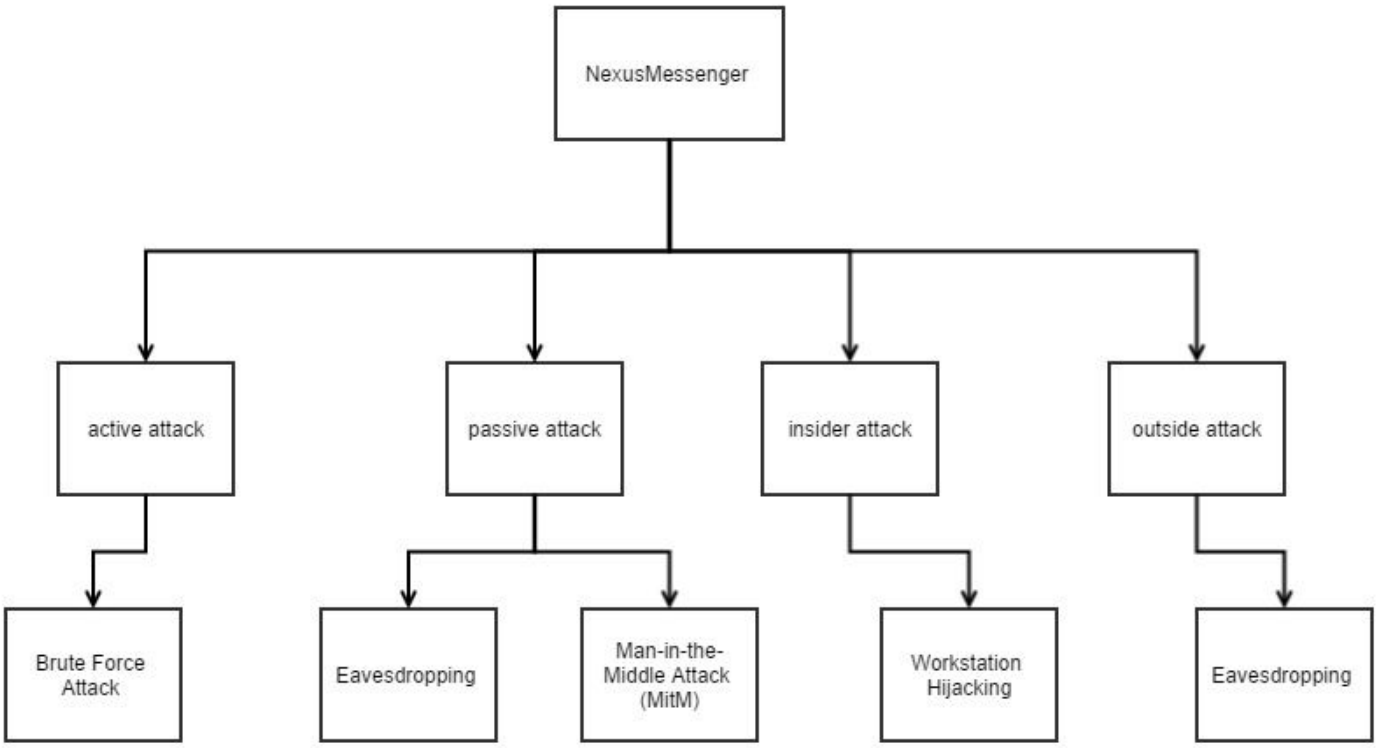
Furthermore, we are choosing to encrypt our messages with Signal Protocol, which combines the Double Ratchet Algorithm, prekeys, and a 3-Diffie-Hellman (DH) handshake. The Double Ratchet Algorithm allows for end-to-end encryption for instant messaging by managing the ongoing renewal and maintenance of shot session keys after an initial key exchange; combines a cryptographic ratchet based on the Diffie-Hellman key exchange and a ratchet based key derivation function. We will protect our system from MitM attack by using this protocol. Since Signal uses Curve25519, AES-256, and HMAC-SHA256, our system is likely to be protected from brute force attack for about 15 years, according to Signal.

For encrypting the messages that are being relaying from client to server, OpenPGP will be used because it uses a fast algorithm to encrypt entire messages. It does this by using the public to encrypt the shorter key that is used to encrypt the entire message. The algorithm used generates a hash code from the user's name and other signature information. Thus, making OpenPGP an efficient authentication system for our platform.

Even though Signal does provide their own version of SSL/TLS by the same creator called Noise Pipes, we will still be using the original OpenSSL and TLS 1.2 to provide modularity and evolutivity which is needed in HTTP. Since TLS and SSL are ubiquitous, we will be using it to provide encrypted communication between the client and web server.

Lastly, we will be using LetsEncrypt to create a HTTPs server with a browser-trusted certificate automatically obtained. It does this by using our server administrator's public key and generating a new key pair when first interacting with it. The agent must prove to LetsEncrypt that they control the key pair by signing it with our private key. Once all of the LetsEncrypt's challenges are completed and everything checks out, it's easy to request, renew and revoke certificates using LetsEncrypt.

Attack Tree


NexusMessenger
active attack
passive attack
insider attack
outside attack
Brute Force Attack
Eavesdropping
Man-in-the-Middle Attack (MitM)
Workstation Hijacking
Eavesdropping